TITLE OF THE INVENTION

IMAGE PROCESSING METHOD, IMAGE PROCESSING APPARATUS, AND IMAGE PROCESSING SYSTEM

BACKGROUND OF THE INVENTION

Field of the Invention

[0001] The present invention relates to an image processing method, an image processing apparatus, and an image processing system.

Description of the Related Art

[0002] Generally, in image output devices, such as laser beam printers, which output high quality images, a device-dependent bit-map is created as follows. Upon receiving a rendering command from an operating system (OS), a multi-level bit map having total of 24 bits for red (R), green (G), and blue (B) is expanded in a multi-level bit-map area. Then, when the processing of all the rendering commands is completed, color processing (color correction, color conversion, n-level processing, etc.) is performed on the entire multi-level bit-map area, thereby forming a device-dependent bit map in a color space.

[0003] However, the expanded multi-level bit map does not possess attribute information for each pixel, and a

determination as to for which type of object (character, graphics, or image) a certain pixel is used cannot be made, or it cannot be determined whether such a pixel forms an edge portion or a central portion of the object. Accordingly, all the pixels unconditionally undergo substantially the same color processing, and such processing is not always optimal for the individual pixels. The concept of such color processing is shown in Fig. 12.

[0004] Accordingly, techniques for providing attribute information for all the pixels in order to achieve optimal color processing on each pixel are known.

[0005] For example, the simplest technique is to reserve extra eight bits for providing attribute information per pixel which consists of 24 RGB bits, and one pixel is represented by RGBα having a total of 32 bits.

[0006] Another technique is to reserve memory for storing attribute information in a different area while maintaining one pixel as 24 RGB bits. The concept of color processing using such an attribute-information memory is shown in Fig. 13. In this technique, the attribute-information storage area is updated every time the corresponding pixel in the multi-level bit-map area is updated.

[0007] Accordingly, when a multi-level bit map is expanded into a multi-level bit-map area in response to a rendering command, attribute information is provided for all

the pixels. Then, optimal color processing can be performed while considering the attribute information for each pixel.

[0008] According to the above-described techniques for providing attribute information, optimal color processing can be performed without presenting any problem as long as new rendering objects simply overwrite old rendering objects. However, when a rendering object which requires a logical operation is processed, attribute information cannot be precisely stored.

[0009] This problem is explained in more detail below with reference to Fig. 14. Fig. 14 illustrates an image object and a graphics object being overlapped when they are expanded into a bit map. Concerning the graphics object, a logical operation, i.e., "rendering is performed only when the graphics object does not have an underlayer" is designated for graphics-image attribute information 102. In this case, in a device-dependent bit-map area 200 in which a bit map is expanded, optimal color processing for images must be performed on the overlapping portion of the image object and the graphics object.

[0010] According to the above-described known attribute-information storage technique, however, after image-object attribute information 101 is stored, the graphics-object attribute information 102 may overwrite the image-object attribute information 101, as indicated in an attribute-

information storage area 100 in Fig. 14. Then, when performing color processing, pixels which actually possess an image attribute, are erroneously determined to possess the graphics attribute, thereby failing to perform optimal rendering processing.

SUMMARY OF THE INVENTION

[0011] Accordingly, in order to solve the above-described problems, it is an object of the present invention to provide an image processing method, an image processing apparatus, and an image processing system in which high quality images can be obtained by performing optimal processing for attributes for the individual pixels even when a rendering operation which requires a logical operation is performed.

[0012] In order to achieve the above object, according to one aspect of the present invention, there is provided an image processing method for creating bit-map data and attribute information for each pixel corresponding to the bit-map data by expanding a rendering command. The image processing method includes: an operation determining step of determining the type of operation to be performed on the attribute information based on logical operation processing specified for the rendering command; a logical operation

processing creating step of creating the logical operation processing for the attribute information based on the determined type of operation; a logical operation processing step of creating the attribute information by executing the logical operation processing; and an inversion step of inverting the attribute information when the attribute information possesses an inversion attribute.

[0013] According to another aspect of the present invention, there is provided an image processing apparatus for creating bit-map data and attribute information for each pixel corresponding to the bit-map data by expanding a rendering command. The image processing apparatus includes an operation determining unit for determining the type of operation to be performed on the attribute information based on logical operation processing specified for the rendering command. A logical operation processing creating unit creates the logical operation processing for the attribute information based on the determined type of operation. A logical operation processor creates the attribute information by executing the logical operation processing. An inversion unit inverts the attribute information when the attribute information possesses an inversion attribute. A color processor performs color processing on the bit-map data based on the attribute information.

[0014] According to still another aspect of the present

invention, there is provided an image processing system including an image processing apparatus and an image forming apparatus connected to each other. The image processing apparatus includes: a bit-map creating unit for creating bit-map data by expanding a rendering command; an attribute-information creating unit for creating attribute information for each pixel corresponding to the bit-map data; a color processor for performing color processing on the bit-map data based on the attribute information; and output unit for outputting the color-processed bit-map data to the image forming apparatus. The attribute-information creating unit includes: an operation determining unit for determining the type of operation to be performed on the attribute information based on logical operation processing specified for the rendering command; a logical operation processing creating unit for creating the logical operation processing for the attribute information based on the determined type of operation; a logical operation processor for creating the attribute information by executing the logical operation processing; and an inversion unit for inverting the attribute information when the attribute information possesses an inversion attribute.

[0015] According to a further aspect of the present invention, there is provided a control program for performing image processing for creating bit-map data and

attribute information for each pixel corresponding to the bit-map data by expanding a rendering command. The control program includes: an operation determining step code for determining the type of operation to be performed on the attribute information based on logical operation processing specified for the rendering command; a logical operation processing creating step code for creating the logical operation processing for the attribute information based on the determined type of operation; a logical operation processing step code for creating the attribute information by executing the logical operation processing; and an inversion step code for inverting the attribute information when the attribute information possesses an inversion attribute.

[0016] According to the present invention, a recording medium storing the above-described control program is also provided.

BRIEF DESCRIPTION OF THE DRAWINGS

[0017] Fig. 1 is a block diagram illustrating the configuration of an information processing system according to an embodiment of the present invention.

[0018] Fig. 2 illustrates the concept of printing processing performed in the information processing system

shown in Fig. 1.

[0019] Fig. 3 is a block diagram illustrating the functional configuration of a driver used in the information processing system shown in Fig. 1.

[0020] Fig. 4 illustrates an example of the format of attribute information used in the information processing system shown in Fig. 1.

[0021] Fig. 5 illustrates the operations to be performed when storing the attribute information.

[0022] Fig. 6 illustrates general logical operation processing.

[0023] Fig. 7A illustrates examples of operation results when the bit patterns indicate a color other than white.

[0024] Fig. 7B illustrates a method for determining the type of operation to be performed on target attribute bits.

[0025] Fig. 7C illustrates a method for creating the attribute information source.

[0026] Fig. 7D is a method for determining the type of operation to be performed on non-target attribute bits.

[0027] Fig. 8A illustrates a method for creating the attribute information pattern.

[0028] Fig. 8B illustrates examples of attribute-information logical operation code.

[0029] Fig. 8C illustrates a method for assigning the operations to the attribute-information logical operation

code.

[0030] Fig. 9 illustrates inversion processing on the attribute information.

[0031] Fig. 10 schematically illustrates color processing performed by referring to the attribute information for multi-level bit maps.

[0032] Fig. 11A is a flowchart illustrating the overall printing processing performed in the information processing system shown in Fig. 1.

[0033] Fig. 11B is a flowchart illustrating details of attribute-information storage processing shown in Fig. 11A.

[0034] Fig. 12 illustrates the concept of the expansion of a multi-level bit map and the color processing for rendering objects using a known technique.

[0035] Fig. 13 illustrates the concept of color processing using an attribute information memory separately from a multi-level bit-map memory according to another known technique.

[0036] Fig. 14 illustrates an example in which optimal color processing has failed by the use of the attribute-information memory according to the known technique shown in Fig. 13.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0037] Figs. 1, 2, and 3 are block diagrams illustrating the configuration of an information processing system employing an information processing method of an embodiment of the present invention.

[0038] In Fig. 1, a central processing unit (CPU) 1 reads a control program implementing the image processing method of this embodiment and corresponding related data stored in a storage medium, such as a floppy disk (FD), a compact disc read only memory (CD-ROM), or an integrated circuit (IC) memory card, by using a media reader 6 connected to the image processing system. Then, based on the control program, the CPU 1 processes image information input through an input device 4 via a system program or an application program loaded from an auxiliary storage device 3 into a main storage device 2, and then outputs the processed image information to an output device 5 or a printer 7. The image information may be stored in a storage medium read by the auxiliary storage device 3 or the media reader 6.

[0039] In this embodiment, the output device 5 is formed of a display device or a communication interface device, and is thus different from the printer 7. The input device 4 is formed of a keyboard, a pointing device, etc. The auxiliary storage device 3 is a single device, such as a hard disk drive or a magneto-optical disk drive, or may be a combination of such drives. The above-described devices may

be connected to each other via a network.

[0040] Fig. 2 illustrates the concept of printing processing from when the control program and the related data are read by the CPU 1 through the media reader 6 until when printing data is sent to the printer 7 based on a printing command input via the input device 4. The printing processing functions under the control of an OS 20 in combination with an application 21 and a driver 22.

[0041] Fig. 3 is a block diagram, together with a flowchart, illustrating the functional configuration of the driver 22 shown in Fig. 2. In step S221, a rendering command is received from the OS 20, and is expanded into a multi-level bit map, which is then stored in a multi-level bit-map area 230. Simultaneously, in step S222, attribute information indicating attributes of a rendering object is stored in an attribute-information storage area 231. The attribute information is information required for performing optimal color processing, for example, the type of object (image, graphics, or text) or edge information indicating the edge of the object.

[0042] Fig. 3 illustrates an example in which the multi-level bit map (24 RGB bits) and the attribute information (8 α bits) are stored in the different areas. However, they may be stored together in a single bit map (32 RGBα bits), in which case, advantages similar to those achieved by

storing the multi-level bit map and the attribute information separately can be obtained.

[0043] In step S223, upon completing the processing of all the rendering commands, the multi-level bit map is combined with the attribute information, and multi-level bit-map data is extracted according to the type of attribute. Then, in step S224, color processing suitable for each type of attribute is performed on the extracted bit-map data, and the individual items of the bit-map data are combined, thereby creating a device-dependent bit map 232. In step S225, the device-dependent bit map 232 is output to the printer 7.

[0044] In this embodiment, the expansion of rendering commands into a bit map and the color processing are performed in the driver 22 of the printer 7 in the information processing system. In the present invention, however, such processing may be performed in the printer 7.

[0045] Details of the processing of this embodiment in the information processing system are given below with reference to Figs. 4 through 10.

[0046] Fig. 4 illustrates the format of attribute information stored in the attribute-information storage area 231 for each pixel of the multi-level bit map. In this embodiment, the attribute information indicates the type of object (a total of three types: image, graphics, and text),

and attribute information for one pixel is formed of at least four bits including three bits representing the three types of objects and one bit representing an inversion attribute, which is described below. Alternatively, extra four bits may be added, resulting in eight bits per pixel (the extra four bits are not used). The attribute-information storage area 231 does not have to be arranged in the order of bits, and may be linearly ordered.

[0047] Fig. 5 illustrates the various operations to be performed when storing attribute information in processing a rendering object which requires a logical operation.

[0048] In this embodiment, one of the following four operations is performed so as to store the attribute information for any type of rendering object which requires a logical operation in the attribute-information storage area 231:

Operation 1: storing attribute information (setting the bit of a targeted attribute);

Operation 2: clear (clearing all the bits of the attribute information);

Operation 3: no change (maintaining all the bits of the attribute information); and

Operation 4: inversion (inverting all the bits of the attribute information).

[0049] Fig. 5 illustrates specific examples of the

individual operations performed on the attribute information in which the bit indicating the graphics attribute is already set when the pixel to be processed forms an underlayer. In operation 1, a new image attribute is stored. In Fig. 5, "ON" indicates that a corresponding bit is set. According to operation 1, by storing the image attribute in the attribute-information storage area 231, the graphics attribute, which forms an underlayer, is cleared.

[0050] A description is given below, with reference to Figs. 6, 7, and 8, of a method for determining which of the four types of operations is to be used when storing attribute information in processing a rendering object which requires a logical operation.

[0051] For simplifying the description, an overview of logical operation processing is discussed below with reference to Fig. 6. Generally, a logical operation indicates an operation result based on the bits indicating the pattern, the source, and the destination (hereinafter simply referred to as "dest"), and the operation result (hereinafter simply referred to as the "result") is stored in the dest.

[0052] The result rxxx (each x indicates the value of the pattern, the source, or the dest (0 or 1)) is 0 or 1 according to the types of combinations (eight) of the pattern, the source, and the dest. That is, in this case, a

total of 256 (2^8) types of logical operations are possible. The logical operation code uniquely determines the type of logical operation. The eight patterns of the results shown in Fig. 6, i.e., r000, r001, r010, r011, r100, r101, r110, r111, are arranged from the lower bit to the upper bit, resulting in the logical operation code. In the example shown in Fig. 6, the source copy is 11001100 (=0xCC), and the pattern copy is 11110000 (=0xF0).

[0053] In the logical operation, according to the source copy (overwriting of the source on the dest), the bits represented by the source are directly reflected as the result (dest) regardless of the combination of the pattern and the dest. According to the pattern copy (overwriting the pattern on the dest), the bits represented by the pattern are directly reflected as the result (dest).

[0054] Figs. 7A through 7D illustrate examples in which the type of operation to be performed on attribute information (hereinafter sometimes simply referred to as the "attribute information operation") is determined based on logical operation processing (logical operation code) specified for a rendering object in this embodiment.

[0055] As discussed above, the logical operation processing is specified by a logical operation code. Any logical operation code may be used as long as it uniquely determines the result, as shown in Fig. 6. In this example,

a logical operation code provided by Windows (Microsoft Corporation) is used. In this code, 0 indicates that the bit is on (ON), and 1 indicates that the bit is off (OFF).

[0056] The operation executed on the bit of the corresponding attribute, which should be turned on if a logical operation is not specified for the attribute information, is described below with reference to Figs. 7A and 7B.

[0057] If the pattern accompanying the rendering object indicates white, the result r1xx shown in Fig. 6 is the bit used for determining the operation performed on the attribute information. In contrast, if the pattern value indicates a color other than white, the result r0xx is the bit to be used. This is because the pattern corresponding to white (0xFFFFFFFF in the RGB value) does not exist in the logical operation processing. Fig. 7A illustrates the result r0xx when the pattern is a color other than white. The result r0xx is equal to the pattern value 0 in the general logical operation processing shown in Fig. 6.

[0058] Fig. 7B illustrates an example in which the attribute information operation is determined based on the result r0xx shown in Fig. 7A. A description is given below of a method for determining the operation based on the result rxxx including both the results r0xx and r1xx.

[0059] A combination of rx00 and rx01 and a combination

of rx10 and rx11 are checked. That is, there are four combinations to be taken by (rx00, rx01) and (rx10, rx11), i.e., (0,0), (0,1), (1,0), and (1,1). These four combinations are assigned to the four operations shown in Fig. 5 as follows:

(0,0) → bit is ON → operation 1 (store attribute information);

(1,1) → bit is OFF → operation 2 (clear);

(0,1) → bit is the same as dest → operation 3 (maintained (no change)); and

(1,0) → bit is inverted from the dest → operation 4 (inversion).

One of the above operations is uniquely determined when the source bit is 0 (rx00, rx01) and when the source bit is 1 (rx10, rx11).

[0060] More specifically, in the example shown in Fig. 7B, when the source bit is 0 (ON), Target_SrcON is obtained as the bit operation performed on the target attribute (attribute information operation). When the source bit is 1 (OFF), Target_SrcOFF is obtained in a manner similar to Target_SrcON. Target_SrcON or the Target SrcOFF results in any one of the above-described four combinations (0,0), (0,1), (1,0), and (1,1), thereby determining the operation to be performed.

[0061] When logical operation processing is not specified

for a rendering command, the operation is determined assuming that a logical operation code indicating simple overwrite processing is specified.

[0062] However, in order to execute the attribute information operation determined as described above, it is necessary to convert the rendering object source into the attribute information source. More specifically, concerning pixels of the rendering object provided with the source, if the pixel patterns indicate white, all the attributes of the corresponding attribute information source are turned OFF (1 in Windows). Conversely, if the pixel patterns indicate a color other than white, all the attributes of the attribute information source are turned ON (0 in Windows). The above-described conversion of the source is shown in Fig. 7C.

[0063] The above-described attribute information operation is performed for the bits of the target attributes (target attribute bits). The operations for the bits of the non-target attributes are uniquely determined as follows according to the determined operations of the target attribute bits.

Target attribute bits	Non-target attribute bits
Operation 1 (store attribute information)	Operation 2 (clear)
Operation 2 (clear)	Operation 2 (clear)
Operation 3 (maintained)	Operation 3 (maintained)
Operation 4 (inversion)	Operation 4 (inversion)

[0064] Fig. 7D illustrates an example in which the operations for non-target attribute bits are determined. As in Fig. 7B, when the source bit is 0 (ON), Target_SrcON is obtained as the target-attribute bit operation, and when the source bit is 1 (OFF), Target_SrcOFF is obtained. Then, according to whether the target attribute bit operation is Target_SrcON or Target_SrcOFF, the non-target attribute bit operation NonTarget_SrcON/NonTarget_SrcOFF is determined as one of the operations 2 through 4, but not operation 1.

[0065] Concerning a rendering object which requires logical operation processing, the above-described operation (one of the operations 1 through 4) is performed on the attribute information source for each attribute, and inversion processing shown in Fig. 9 is further performed, thereby obtaining precise attribute information.

[0066] In this embodiment, logical operation processing is created based on the corresponding one of the operations 1 through 4 for the attribute information rather than simply

executing the operations 1 through 4, which makes the execution of the operation more efficient. That is, when logical operation processing based on one of the above-described operations 1 through 4 is performed, the same results obtained when the operations determined for the target attribute bits and the non-target attribute bits are performed for the attribute information source can be obtained.

[0067] Logical operation processing for the attribute information in this embodiment is discussed below more specifically with reference to Figs. 8A through 8C.

[0068] When performing the operation on the attribute information, it is necessary to switch among the operations 1 through 4 according to the attribute information for each pixel, according to the attribute bit, or according to the attribute information source. In this embodiment, for efficiently performing this switching operation, the individual operations for the attribute information are represented in a format similar to the general logical operation shown in Fig. 6, i.e., a format using the attribute information pattern and the attribute information source. To implement such efficient processing, the attribute information pattern shown in Fig. 8A should be prepared first. In the attribute information pattern shown in Fig. 8A, the bit indicating the attribute of a certain

rendering object (in this example, the image attribute) is turned ON, and the other bits are turned OFF.

[0069] Fig. 8B illustrates an example of a combination of the attribute information pattern shown in Fig. 8A, the attribute information source shown in Fig. 7C, and the attribute information operation shown in Fig. 7D in the general format of the logical operation shown in Fig. 6. That is, Fig. 8B illustrates the logical operation code for the attribute information used in this embodiment. The result in Fig. 8B indicates the logical operation result, and the value (OFF or ON) corresponds to the result rxxxx (0 or 1). Thus, the result of the logical operation is uniquely determined as shown in Fig. 8C according to the target bit operation (Target_SrcON, Target_SrcOFF) shown in Fig. 7B and the non-target bit operation (NonTarget_SrcON, NonTarget_SrcOFF).

[0070] For example, if the attribute information pattern in Fig. 8B is OFF, the result in Fig. 8B is determined as shown in Fig. 8C based on the operation for the non-target attribute bit (NonTarget_SrcON, NonTarget_SrcOFF). That is, if the operation 2 (clear) is set for the non-target attribute bit, the result automatically is turned OFF to clear the bit regardless of the attribute-information storage area (dest). If the operation 3 (maintained) is set for the non-target attribute bit, the bit in the dest (OFF

or ON) is maintained. If the operation 4 (inversion) is set, the result obtained by inverting the dest (ON or OFF) is used. If the attribute information pattern is OFF, operation 1 (store the attribute information) is not set for the non-target attribute bit, and thus, operation 1 need not be considered.

[0071] If the attribute information pattern is ON in Fig. 8B, the result shown in Fig. 8B is determined as shown in Fig. 8C based on the operation for the target attribute bit (Target_SrcOFF, Target_SrcON) shown in Fig. 7B in a manner similar to the case when the attribute information pattern is OFF. When the attribute information pattern is ON, however, the operation 1 can be set for the non-target attribute bit. If the operation 1 is set, the attribute information pattern is ON, and thus, the result is turned ON regardless of the dest (attribute-information storage area).

[0072] In this embodiment, by using the uniquely determined result shown in Fig. 8B as the attribute-information logical operation code, processing can be performed in a manner similar to the general logical operation processing. That is, the result OFF/ON shown in Fig. 8B is substituted with the result rxxx 0/1. Then, operation processing can be executed all at the same time without being aware of the target attribute bits of the attribute information.

[0073] The result obtained by performing the logical operation shown in Fig. 8B is stored in the attribute-information storage area 231 as the attribute information. The attribute information obtained by the logical operation is the same result as that obtained by performing the attribute information operation on each attribute information source.

[0074] Upon completing all the rendering objects, i.e., the logical operation processing for the accompanying attribute information, the bit inversion processing, which is one of the features of this embodiment, is performed. The bit inversion processing is described below with reference to Fig. 9.

[0075] When the processing for all the rendering objects is completed, the bit pattern indicated by the attribute information for each pixel stored in the attribute-information storage area 231 is one of the following four states A through D:

State A: all the bits are OFF (maintained as the initial value (nothing is rendered));

State B: only one attribute is ON (final operation is operation 1 (store attribute information));

State C: only one attribute is OFF (final operation is operation 4 (inversion) after state B));

State D: all the bits are ON (final operation is

operation 4 (inversion) after state A)).

[0076] In the above-described bit patterns, only when the inversion bit is ON in state C or D, all four bits are inverted. In this case, exclusive-OR is performed between the inversion bits and the other attribute bits. Then, processing can be performed simultaneously regardless of the ON/OFF state of the inversion bit or the other attribute bits.

[0077] Fig. 9 illustrates states A through D of the bit patterns stored in the attribute-information storage area 231, and also shows the result obtained by performing inversion processing on the patterns in states C and D. As described above, by performing the above-described inversion processing on the bit patterns stored in the attribute-information storage area 231 after the processing for all the rendering objects is completed, ideal attribute information can be created.

[0078] Fig. 10 is a schematic diagram illustrating an overview of color processing (steps S223 and S224 in Fig. 3) performed for a multi-level bit map expanded into the multi-level bit map area 230 by referring to the attribute information created in the attribute-information storage area 231. In this embodiment, a multi-level bit map is extracted for each attribute (according to the type of object), as shown in Fig. 10, from the multi-level bit-map

area 230 based on the attribute information stored in the attribute-information storage area 231. That is, a graphics multi-level bit map 241, a text multi-level bit map 242, and an image multi-level bit map 243 are extracted.

[0079] Then, color processing suitable for each attribute is performed on the extracted bit map, and the processed results are combined, thereby creating the device-dependent bit map 232. Alternatively, color processing may be performed for each pixel in the multi-level bit-map area 230 by referring to the attribute information rather than extracting the multi-level bit map for each attribute from the multi-level bit-map area 230.

[0080] The above-described color processing is implemented by forming the processing indicated by the flowcharts of Figs. 11A and 11B into a program and by executing this program. The overall printing processing of this embodiment is described in detail with reference to the flowcharts of Figs. 11A and 11B.

[0081] Upon receiving a print execution command from the input device 4, the OS 20 loaded, together with the driver 22 and the application 21, from the auxiliary storage device 3 into the main storage device 2 receives the corresponding message. The OS 20 then sends the print execution message to the currently active application 21.

[0082] The application 21 then converts the print

execution message into a format recognizable by the OS 20, and sends a message including the print data and the command to the OS 20. Then, the OS 20 converts the message into a command recognizable by the driver 22, and sends it to the driver 22. In the driver 22, the processing indicated by the flowcharts shown in Figs. 11A and 11B is executed.

[0083] In step S101, upon receiving an initializing message from the OS 20, the driver 22 allocates the multi-level bit-map area 230 and the attribute-information storage area 231 shown in Fig. 3, and then clears the content in the storage areas 230 and 231.

[0084] Then, in step S102, the driver 22 expands a multi-level bit map according to the rendering command sent from the OS 20, and stores the expanded multi-level bit map in the multi-level bit-map area 230. In step S103, the driver 22 also stores the attributes of the rendering command in the attribute-information storage area 231 as the attribute information.

[0085] The attribute-information storage processing in step S103 is discussed below in detail with reference to Fig. 11B.

[0086] The main feature of this embodiment is to store the attribute information of a rendering command in the attribute-information storage area 231. Thus, in step S201, the type of attribute of the rendering command is determined.

It is then determined in step S202 whether the rendering command pattern is white. If so, the process proceeds to step S203. In step S203, a suitable portion of the logical operation code designated for the rendering command is extracted, and the operation (Target_SrcON/Target_SrcOFF) for the attribute (target attribute bit) is created, as described with reference to Figs. 7A through 7D. If it is found in step S202 that the rendering command pattern is a color other than white, the process proceeds to step S204. In step S204, the operation (Target_SrcON/Target_SrcOFF) for the target attribute bit is created, as in step S203. That is, in step S203 or S204, the operation to be performed when storing the rendering command attribute in the attribute-information storage area 231 is created, as shown in Fig. 7B.

[0087] If logical operation processing is not specified for the rendering command, the corresponding operation is created, assuming that the logical operation code is set as simple overwrite processing.

[0088] Subsequently, in step S205, the operation (NonTarget_SrcON/NonTarget_SrcOFF) for the attributes other than the attribute of the rendering command (non-target attribute bits) is created, as shown in Fig. 7D, based on the operation for the rendering command attribute created in step S203 or S204.

[0089] By performing the operations created as described-

above on all the bits of the attribute information individually, precise attribute information can be created. In this embodiment, operation processing can be performed simultaneously regardless of the content of the individual bits of the attribute information, as described with reference to Figs. 8A through 8C, thereby making the processing more efficient.

[0090] Then, in step S206, the logical operation code for the attribute information is created, as shown in Figs. 8B and 8C, based on the operations created in steps S203 or S204, and S205.

[0091] In step S207, the attribute information source is created based on the rendering object, as shown in Fig. 7C. Subsequently, in step S208, the attribute information pattern is created, as shown in Fig. 8A, in the format in which the bit indicating the attribute of the rendering command is turned on and the other bits are turned off by using the attribute information format shown in Fig. 4. In step S209, the regions corresponding to the individual pixels in the multi-level bit map in the attribute-information storage area 231 are extracted, and are set as the attribute-information dest.

[0092] Then, in step S210, logical operation processing is executed by using the attribute-information logical operation code created in step S209 based on the attribute

information source, the attribute information pattern, and the attribute information dest, as shown in Figs. 8B and 8C. In step S211, the obtained operation result is stored in the attribute information dest.

[0093] By the processing of steps S201 through S211, the attribute-information storage processing in step S103 is completed, and the process returns to step S104 of Fig. 11A.

[0094] A determination is made in step S104 as to whether the expansion of the multi-level bit map and the storage of the corresponding attribute information have been completed for all the rendering commands. If the result of step S104 is yes, the process proceeds to step S105. In step S105, the inversion attribute of the attribute information is converted as shown in Fig. 9, for the inversion bits in the attribute-information storage area 231. As a result, precise attribute information is generated. If the result of step S104 is no, the process returns to step S102.

[0095] In step S106, a multi-level bit map is extracted for each attribute according to the attribute information. In step S107, color processing optimal for the attribute is then performed. Then, in step S108, the device-dependent bit map 232 is created, as shown in Fig. 10.

[0096] By performing the processing in steps S106 through S108, the device-dependent bit maps 232 created for the individual attributes are combined. It is thus possible to

obtain an optimal output result regardless of the type of logical operation processing designated for the rendering command. It is determined in step S109 whether all the attributes have been processed.

[0097] Then, in step S110, the combined device-dependent bit map is transferred to the printer 7. The printing processing in this embodiment is then completed.

[0098] Any type of printer 7 in this embodiment may be used as long as it is able to output the device-dependent bit maps. The printer 7 may be a binding printer or a non-binding printer.

[0099] In this embodiment, the rendering command possesses three types of attributes, i.e., image, text, and graphics. However, by adding the number of bits of the attribute information shown in Fig. 4, the number of attributes to be handled may be increased or decreased.

[0100] As described above, according to this embodiment, it is possible to store precise attribute information for all the pixels including a rendering object which requires logical operation processing. As a consequence, optimal color processing for each pixel can be performed, thereby obtaining a high quality output result.

[0101] In this embodiment, a rendering object is expanded into a multi-level bit map. However, any type of format may be employed as long as it represents color information of

pixels. For example, when binary data is used, the present invention is still applicable.

[0102] The present invention is applicable to a single device (for example, a copying machine or a facsimile machine) or a system consisting of a plurality of devices (for example, a combination of a host computer, an interface, a reader, and a printer).

[0103] The object of the present invention can also be achieved by the following modification. A storage medium (or a recording medium) for storing software program code implementing the functions of the above-described embodiment may be supplied to a system or an apparatus. Then, a computer (or a CPU or an micro processing unit (MPU)) of the system or the apparatus may read and execute the program code from the storage medium (recording medium). In this case, the program code itself read from the storage medium implements the novel functions of the embodiment. Accordingly, the program code itself, and the storage medium storing such program code constitute the present invention.

[0104] The functions of the above-described embodiment may be implemented not only by running the read program code on the computer, but also by wholly or partially executing the processing by an OS running on the computer or in cooperation with other application software based on the instructions of the program code. The present invention

also encompasses such a modification.

[0105] The functions of the embodiment may also be implemented by the following modification. The program code read from the storage medium is written into a memory provided on a feature expansion board inserted into the computer or a feature expansion unit connected to the computer. Then, a CPU provided for the feature expansion board or the feature expansion unit partially or wholly executes processing based on the instructions of the program code.

[0106] While the present invention has been described with reference to what are presently considered to be the preferred embodiment, it is to be understood that the invention is not limited to the disclosed embodiment. On the contrary, the invention is intended to cover various modifications and equivalent arrangements included within the spirit and scope of the appended claims. The scope of the following claims is to be accorded the broadest interpretation so as to encompass all such modifications and equivalent structures and functions.